Computational Complexity Theory

Lecture 8: Polynomial Hierarchy

Department of Computer Science,
Indian Institute of Science

Problems between NP & PSPACE

- There are decision problems that don't appear to be captured by nondeterminism alone (i.e., with a single ∃ or ∀ quantifier), unlike problems in NP and co-NP.

- Example.

  Eq-DNF = {(ϕ,k): ϕ is a **DNF** and there's a DNF ψ of size $\leq$ k that is equivalent to ϕ}

- Two Boolean formulas on the same input variables are *equivalent* if their evaluations agree on every assignment to the variables.

Problems between NP & PSPACE

- There are decision problems that don't appear to be captured by nondeterminism alone (i.e., with a single ∃ or ∀ quantifier), unlike problems in NP and co-NP.

- Example.

  Eq-DNF = {(ϕ,k): ϕ is a **DNF** and there's a DNF ψ of size ≤ k that is equivalent to ϕ}

- Is Eq-DNF in NP? …if we give a DNF ψ as a certificate, it is not clear how to efficiently verify that ψ and ϕ are equivalent. (W.l.o.g. k ≤ size of ϕ .)

Class Σ_2

- Definition. A language L is in Σ_2 if there's a polynomial function q(.) and a poly-time TM M (the "verifier") s.t.

$x \in L \iff \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v) = 1$.

Class Σ_2

- **Definition.** A language L is in Σ_2 if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)} \ \forall v \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x,u,v) = 1.$$

- **Obs.** Eq-DNF is in Σ_2.
- **Proof.** Think of u as another DNF ψ and v as an assignment to the variables. Poly-time TM M checks if ψ has size $\leq k$ and $\phi(v) = \psi(v)$.

Class Σ_2

- **Definition.** A language L is in Σ_2 if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

  $x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v) = 1$.

- **Obs.** Eq-DNF is in Σ_2.
- **Proof.** Think of u as another DNF ψ and v as an assignment to the variables. Poly-time TM M checks if ψ has size $\leq k$ and $\phi(v) = \psi(v)$.

- **Remark.** *(Masek 1979)* Even if ϕ is given by its truth-table, the problem (i.e., DNF-MCSP) is NP-complete.

Class Σ_2

- Definition. A language L is in Σ_2 if there's a polynomial function q(.) and a poly-time TM M (the "verifier") s.t.

$x \in L \iff \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v) = 1$.

- Another example.

Succinct-SetCover = $\{(\phi_1,\ldots\phi_m,k)$: ϕ_i's are DNFs and there's an $S \subseteq [m]$ of size $\leq k$ s.t. $\vee_{i\in S}\ \phi_i$ is a tautology$\}$

Class Σ_2

- Definition. A language L is in Σ_2 if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

  $x \in L \Longleftrightarrow \exists u \in \{0,1\}^{q(|x|)} \ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v) = 1$.

- Obs. *(Homework)* Succinct-SetCover is in Σ_2.

Class Σ_2

- Definition. A language L is in Σ_2 if there's a polynomial function q(.) and a poly-time TM M (the "verifier") s.t.

  $x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v) = 1$.

- Obs. *(Homework)* Succinct-SetCover is in Σ_2.

- Other natural problems in PH: *"Completeness in the Polynomial-Time Hierarchy: A Compendium"* by *Schaefer and Umans (2008)*.

Class Σ_2

- **Definition.** A language L is in Σ_2 if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

  $x \in L \iff \exists u \in \{0,1\}^{q(|x|)} \ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v) = 1$.

- **Obs.** $P \subseteq NP \subseteq \Sigma_2$.

Class $\sum_i$

- **Definition.** A language L is in $\sum_i$ if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

  $$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \ \forall u_2 \in \{0,1\}^{q(|x|)} \ Q_i u_i \in \{0,1\}^{q(|x|)}$$

  $$\text{s.t. } M(x,u_1,\ldots,u_i) = 1,$$

  where Q_i is $\exists$ or $\forall$ if i is odd or even, respectively.

- **Obs.** $\sum_i \subseteq \sum_{i+1}$ for every i.

Polynomial Hierarchy

- **Definition.** A language L is in Σ_i if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

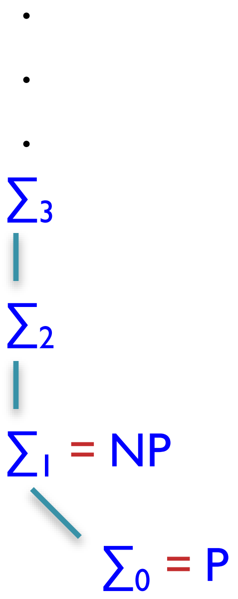
$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)}\ \ \forall u_2 \in \{0,1\}^{q(|x|)}\ \ Q_i u_i \in \{0,1\}^{q(|x|)}$$

$$\text{s.t. } M(x, u_1, \ldots, u_i) = 1,$$

where Q_i is $\exists$ or $\forall$ if i is odd or even, respectively.

- **Definition.** *(Meyer & Stockmeyer 1972)*

$$PH = \bigcup_{i \in \mathbf{N}} \Sigma_i\ .$$



Class $\prod_i$

- Definition. $\prod_i$ = co-$\sum_i$ = $\{ L : \overline{L} \in \sum_i \}$.

- Obs. A language L is in $\prod_i$ if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

  $$x \in L \Leftrightarrow \forall u_1 \in \{0,1\}^{q(|x|)} \ \exists u_2 \in \{0,1\}^{q(|x|)} \ Q_i u_i \in \{0,1\}^{q(|x|)}$$

  s.t. $M(x, u_1, \ldots, u_i) = 1$,

  where Q_i is $\forall$ or $\exists$ if i is odd or even, respectively.

Class Π_i

- **Definition.** $\Pi_i = \text{co-}\Sigma_i = \{ L : \overline{L} \in \Sigma_i \}$.

- **Obs.** A language L is in Π_i if there's a polynomial function $q(.)$ and a poly-time TM M (the "verifier") s.t.

  $$x \in L \Leftrightarrow \forall u_1 \in \{0,1\}^{q(|x|)} \ \exists u_2 \in \{0,1\}^{q(|x|)} \ \ Q_i u_i \in \{0,1\}^{q(|x|)}$$

  $$\text{s.t. } M(x,u_1,\ldots, u_i) = 1,$$

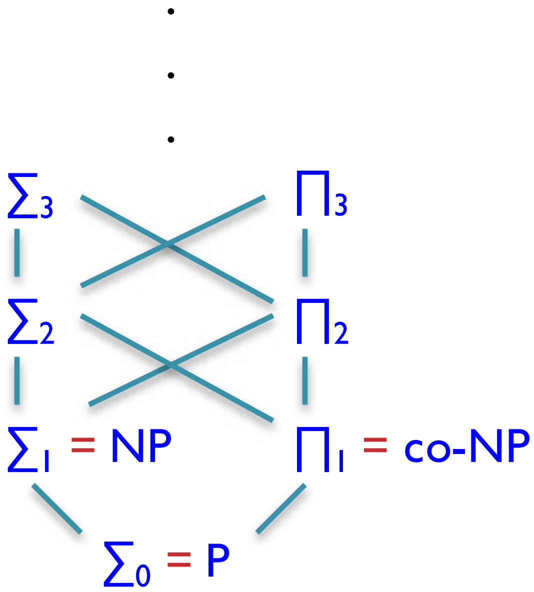
  where Q_i is $\forall$ or $\exists$ if i is odd or even, respectively.

- **Obs.** $\Sigma_i \subseteq \Pi_{i+1} \subseteq \Sigma_{i+2}$.

Polynomial Hierarchy

- Obs. PH $= \bigcup_{i \in \mathbf{N}} \Sigma_i = \bigcup_{i \in \mathbf{N}} \Pi_i$.

PH =



$\vdots$

Σ_3 Π_3

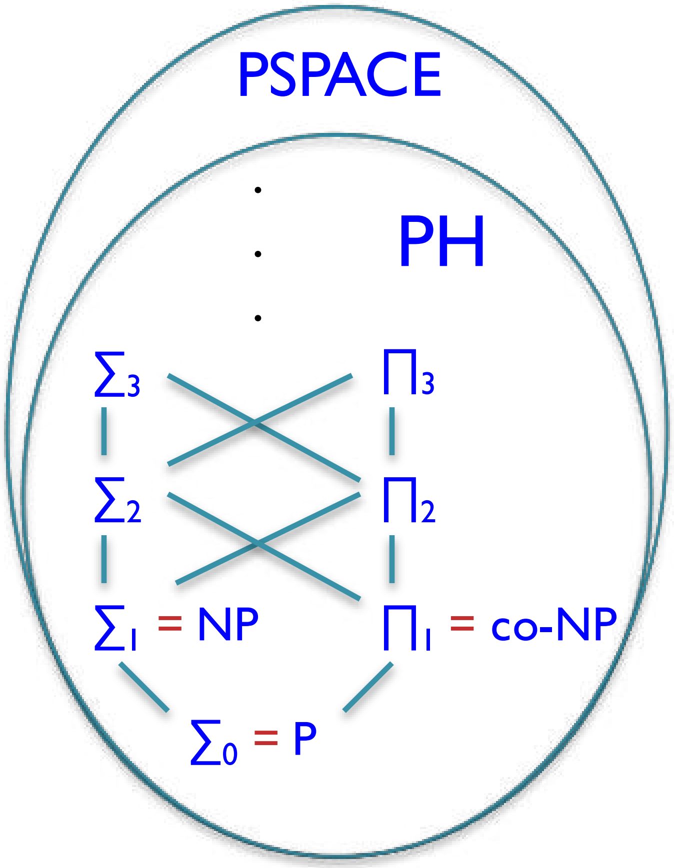
Σ_2 Π_2

Σ_1 = NP Π_1 = co-NP

Σ_0 = P

Polynomial Hierarchy

- **Claim.** PH ⊆ PSPACE .
- **Proof.** Similar to the proof of TQBF ∈ PSPACE.



PSPACE

PH

.
.
.

Σ_3 Π_3

Σ_2 Π_2

Σ_1 = NP

Π_1 = co-NP

Σ_0 = P

Does PH collapse?

- General belief. Just as many of us believe $P \neq NP$ (i.e. $\Sigma_0 \neq \Sigma_1$) and $NP \neq co\text{-}NP$ (i.e. $\Sigma_1 \neq \Pi_1$), we also believe that for every i, $\Sigma_i \neq \Sigma_{i+1}$ and $\Sigma_i \neq \Pi_i$.

- Definition. We say PH collapses to the i-th level if $\Sigma_i = \Sigma_{i+1}$. (justified in the next theorem)

- Conjecture. There is no i such that PH collapses to the i-th level.

Does PH collapse?

- General belief. Just as many of us believe P ≠ NP (i.e. $\Sigma_0 \neq \Sigma_1$) and NP ≠ co-NP (i.e. $\Sigma_1 \neq \Pi_1$), we also believe that for every i, $\Sigma_i \neq \Sigma_{i+1}$ and $\Sigma_i \neq \Pi_i$.

- Definition. We say PH collapses to the i-th level if $\Sigma_i = \Sigma_{i+1}$. (justified in the next theorem)

- Conjecture. There is no i such that PH collapses to the i-th level.

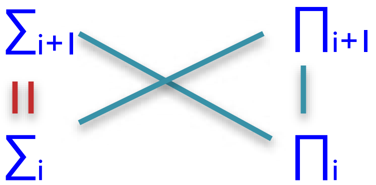
This is stronger than the P ≠ NP conjecture.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.

PH collapse theorems

- **Theorem.** If $\Sigma_i = \Sigma_{i+1}$ then $\mathrm{PH} = \Sigma_i$.
- **Proof.**



PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $\mathrm{PH} = \Sigma_i$.
- Proof.

$$\begin{tikzcd}
\Sigma_{i+1} \arrow[d, equal] \arrow[dr, dash] & \Pi_{i+1} \arrow[d, equal] \arrow[dl, dash] \\
\Sigma_i & \Pi_i
\end{tikzcd}$$

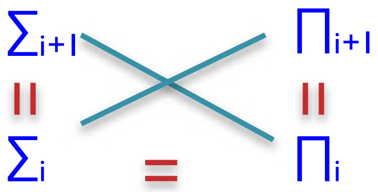
PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof.

$$\begin{tikzcd}
\Sigma_{i+1} \arrow[d, equal] \arrow[dr, dash] & \Pi_{i+1} \arrow[d, equal] \arrow[dl, dash] \\
\Sigma_i & \Pi_i \arrow[l]
\end{tikzcd}$$

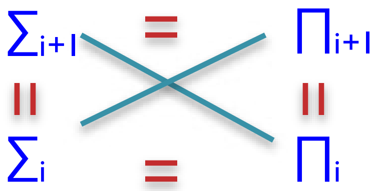
PH collapse theorems

- **Theorem.** If $\Sigma_i = \Sigma_{i+1}$ then $\mathrm{PH} = \Sigma_i$.
- **Proof.**



PH collapse theorems

- **Theorem.** If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- **Proof.**



$$\begin{tikzcd}
\Sigma_{i+1} \arrow[r, phantom, "="] \arrow[d, phantom, "\parallel"] \arrow[dr, dash] & \Pi_{i+1} \arrow[d, phantom, "\parallel"] \\
\Sigma_i \arrow[r, phantom, "="] \arrow[ur, dash] & \Pi_i
\end{tikzcd}$$

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $\mathrm{PH} = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function q(.) and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

- Define $L' = \{(x, u_1) : \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(\underbrace{x, u_1}_{}, \ldots, u_{i+2}) = 1\}$

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

- Clearly, L' is in $\Pi_{i+1} = \Sigma_i$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

- Also, $x \in L \Leftrightarrow \exists u_1$ s.t. $(x, u_1) \in L'$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

- Also, $x \in L \Leftrightarrow \exists u_1 \exists v_1 \forall v_2 \ldots Q_i v_i$ s.t. $N(x, u_1, v_1 \ldots, v_i) = 1$, where N is a poly-time TM.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $\mathrm{PH} = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

- Also, $x \in L \Leftrightarrow \underbrace{\exists u_1 \, \exists v_1}_{\text{Merge the quantifiers}} \forall v_2 \ldots Q_i v_i$ s.t. $N(x, u_1, v_1 \ldots, v_i) = 1$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

- Also, $x \in L \Leftrightarrow \exists v'_1 \forall v_2 \ldots Q_i v_i$ s.t. $N(x, v'_1 \ldots, v_i) = 1$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Sigma_{i+1}$ then $PH = \Sigma_i$.
- Proof. Hence $\Sigma_i = \Sigma_{i+1} = \Pi_i = \Pi_{i+1}$.

  Goal is to show that $\Sigma_{i+1} = \Sigma_{i+2}$.

- Let L be a language in Σ_{i+2} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+2} u_{i+2}$ s.t. $M(x, u_1, \ldots, u_{i+2}) = 1$.

- Hence, L is a language in $\Sigma_i = \Sigma_{i+1}$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $\mathrm{PH} = \Sigma_i$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.
- Proof. Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

PH collapse theorems

- **Theorem.** If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.
- **Proof.** Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

$x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(x, u_1, \ldots, u_{i+1}) = 1$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.
- Proof. Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(x, u_1, \ldots, u_{i+1}) = 1$.

- Define $L' = \{(x, u_1): \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(\underbrace{x, u_1}_{}, \ldots, u_{i+1}) = 1\}$

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.
- Proof. Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function q(.) and a poly-time TM M s.t.

$x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(x, u_1, \ldots, u_{i+1}) = 1$.

- Clearly, L' is in $\Pi_i = \Sigma_i$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.
- Proof. Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(x, u_1, \ldots, u_{i+1}) = 1$.

- Also, $x \in L \Leftrightarrow \exists u_1$ s.t. $(x, u_1) \in L'$.

PH collapse theorems

- **Theorem.** If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.
- **Proof.** Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(x, u_1, \ldots, u_{i+1}) = 1$.

- Also, $x \in L \Leftrightarrow \exists u_1 \exists v_1 \forall v_2 \ldots Q_i v_i$ s.t. $N(x, u_1, v_1 \ldots, v_i) = 1$, where N is a poly-time TM.

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $\mathrm{PH} = \Sigma_i$.
- Proof. Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1} \quad \text{s.t.} \quad M(x, u_1, \ldots, u_{i+1}) = 1.$$

- Also, $x \in L \Leftrightarrow \underbrace{\exists u_1 \exists v_1}_{\text{Merge the quantifiers}} \forall v_2 \ldots Q_i v_i$ s.t. $N(x, u_1, v_1 \ldots, v_i) = 1$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $PH = \Sigma_i$.
- Proof. Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(x, u_1, \ldots, u_{i+1}) = 1$.

- Also, $x \in L \Leftrightarrow \exists v'_1 \forall v_2 \ldots Q_i v_i$ s.t. $N(x, v'_1, \ldots, v_i) = 1$.

PH collapse theorems

- Theorem. If $\Sigma_i = \Pi_i$ then $\mathrm{PH} = \Sigma_i$.
- Proof. Goal is to show that $\Sigma_i = \Pi_i \Rightarrow \Sigma_i = \Sigma_{i+1}$.

- Let L be a language in Σ_{i+1} . Then there's a polynomial function q(.) and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_{i+1} u_{i+1}$ s.t. $M(x, u_1, \ldots, u_{i+1}) = 1$.

- Hence, L is a language in Σ_i .

Complete problems in PH ?

- Recall, to define completeness of a complexity class, we need an appropriate notion of a *reduction*.
- What kind of reductions will be suitable is guided by *a complexity question*, like a comparison between the complexity class under consideration & another class.
- Is P = PH ? …use poly-time Karp reduction!

- Definition. A language L' is *PH-hard* if for every L in PH, $L \leq_p L'$. Further, if L' is in PH then L' is *PH-complete*.

Complete problems in PH ?

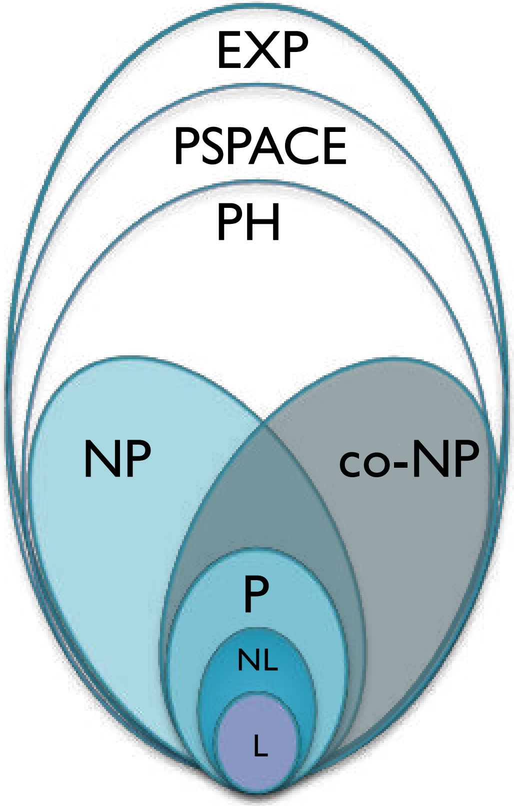
- Fact. If L is poly-time reducible to a language in Σ_i then L is in Σ_i .   (we've seen a similar fact for NP)

Complete problems in PH ?

- **Fact.** If L is poly-time reducible to a language in Σ_i then L is in Σ_i . (we've seen a similar fact for NP)
- **Observation.** If PH has a complete problem then PH collapses.
- **Proof.** If L is *PH-complete* then L is in Σ_i for some i. Now use the above fact to infer that PH = Σ_i .

Complete problems in PH ?

- Fact. If L is poly-time reducible to a language in Σ_i then L is in Σ_i . (we've seen a similar fact for NP)
- Corollary. PH $\subsetneq$ PSPACE unless PH collapses.



Complete problems in $\sum_i$

- Recall, to define completeness of a complexity class, we need an appropriate notion of a *reduction*.
- What kind of reductions will be suitable is guided by *a complexity question*, like a comparison between the complexity class under consideration & another class.
- Is $P = \sum_i$? …use poly-time Karp reduction!

- Definition. A language L' is $\sum_i$ *-hard* if for every L in $\sum_i$, $L \leq_p L'$. Further, if L' is in $\sum_i$ then L' is $\sum_i$ *-complete*.

Complete problems in Σ_i

- Definition. The language Σ_i-SAT contains all *true* QBF with i alternating quantifiers starting with $\exists$.

- Theorem. Σ_i-SAT is Σ_i *-complete*. (Σ_1-SAT is just SAT)

Complete problems in Σ_i

- **Definition.** The language Σ_i-SAT contains all *true* QBF with i alternating quantifiers starting with $\exists$.

- **Theorem.** Σ_i-SAT is Σ_i *-complete*.
- **Proof.** Easy to see that Σ_i-SAT is in Σ_i .

$x = \exists v_1 \forall v_2 \ldots Q_i v_i\ \phi(v_1, \ldots, v_i) \in \Sigma_i\text{-SAT} \quad \Leftrightarrow$

$\exists u_1 \forall u_2 \ldots Q_i u_i$ s.t. $M(x, u_1, \ldots, u_i) = 1$,

where M outputs $\phi(u_1, \ldots, u_i)$.

Complete problems in Σ_i

- Definition. The language Σ_i-SAT contains all *true* QBF with i alternating quantifiers starting with $\exists$.

- Theorem. Σ_i-SAT is Σ_i *-complete*.
- Proof. Let L be a language in Σ_i . Then there's a polynomial function q(.) and a poly-time TM M s.t.

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_i u_i \quad \text{s.t.} \quad M(x, u_1, \ldots, u_i) = 1.$$

Complete problems in Σ_i

- Definition. The language Σ_i-SAT contains all *true* QBF with i alternating quantifiers starting with ∃.

- Theorem. Σ_i-SAT is Σ_i *-complete*.
- Proof. Let L be a language in Σ_i . Then there's a polynomial function q(.) and a poly-time TM M s.t.

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_i u_i \quad \text{s.t.} \quad \underbrace{\phi(x, u_1, \ldots, u_i)}_{\substack{\text{Boolean circuit} \\ (\textit{by Cook-Levin})}} = 1 .$$

Complete problems in $\sum_i$

- **Definition.** The language $\sum_i$-SAT contains all *true* QBF with i alternating quantifiers starting with $\exists$.

- **Theorem.** $\sum_i$-SAT is $\sum_i$ *-complete*.
- **Proof.** Let L be a language in $\sum_i$. Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Leftrightarrow \exists u_1 \forall u_2 \ldots Q_i u_i \ \phi(x, u_1, \ldots, u_i)$ is true .

Complete problems in Σ_i

- **Definition.** The language Σ_i-SAT contains all *true* QBF with i alternating quantifiers starting with $\exists$.

- **Theorem.** Σ_i-SAT is Σ_i *-complete*.
- **Proof.** Let L be a language in Σ_i . Then there's a polynomial function $q(.)$ and a poly-time TM M s.t.

$$x \in L \iff \exists u_1 \forall u_2 \ldots Q_i u_i \;\; \phi(x, u_1, \ldots, u_i) \in \Sigma_i\text{-SAT} .$$

Other complete problems in Σ_2

- Ref. *“Completeness in the Polynomial-Time Hierarchy: A Compendium”* by *Schaefer and Umans (2008)*.

- Theorem. Eq-DNF and Succinct-SetCover are Σ_2 *-complete*.

An alternate characterization of PH

Oracle definition of Σ_i

- **Definition.** A language L is in $NP^{\Sigma_i\text{-SAT}}$ if there is a poly-time NTM with oracle access to Σ_i-SAT that decides L.

- **Theorem.** $\Sigma_{i+1} = NP^{\Sigma_i\text{-SAT}}$.

Oracle definition of Σ_i

- Definition. A language L is in $NP^{\Sigma_i\text{-SAT}}$ if there is a poly-time NTM with oracle access to Σ_i-SAT that decides L.

- Theorem. $\Sigma_{i+1} = NP^{\Sigma_i\text{-SAT}}$.

- Observe that Σ_1-SAT = SAT. We'll prove the special case $\Sigma_2 = NP^{SAT}$. The proof of the theorem is similar.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = \mathrm{NP}^{\mathrm{SAT}}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

$$x \in L \iff \exists u \in \{0,1\}^{q(|x|)} \ \forall v \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x,u,v) = 1.$$

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = \mathrm{NP}^{\mathrm{SAT}}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

$$x \in L \iff \exists u \in \{0,1\}^{q(|x|)} \; \forall v \in \{0,1\}^{q(|x|)} \text{ s.t. } \underbrace{\phi(x,u,v)}_{\substack{\text{Boolean circuit} \\ (\textit{by Cook-Levin})}} = 1 .$$

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $\neg\phi(x,u,v) = 0$.

- Think of a NTM N that has the knowledge of M. On input x, it guesses $u \in \{0,1\}^{q(|x|)}$ non-deterministically and computes the circuit $\phi(x,u,v)$. Then, it queries the SAT oracle with $\neg\phi(x,u,v)$.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = \mathrm{NP}^{\mathrm{SAT}}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

  $x \in L \Longleftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $\neg\phi(x,u,v) = 0$.

- Think of a NTM N that has the knowledge of M. On input x, it guesses $u \in \{0,1\}^{q(|x|)}$ non-deterministically and computes the circuit $\phi(x,u,v)$. Then, it queries the SAT oracle with $\neg\phi(x,u,v)$.
- Wait.. $\neg\phi(x,u,v)$ needn't be a CNF!

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in Σ_2. There's a polynomial function $q(.)$ and a poly-time TM M s.t.

$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)} \ \forall v \in \{0,1\}^{q(|x|)}$ s.t. $\neg\phi(x,u,v) = 0$.

- Think of a NTM N that has the knowledge of M. On input x, it guesses $u \in \{0,1\}^{q(|x|)}$ non-deterministically and computes the circuit $\phi(x,u,v)$. Then, it queries the SAT oracle with $\neg\phi(x,u,v)$.
- Use auxiliary variables to transform the query circuit to a CNF. (*Cook-Levin*)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most <u>one</u> query to SAT oracle on every computation path on input x.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and $u_1, v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:
- M simulates N on input x with w as the non-deterministic choices.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:
- M simulates N on input x with w as the computation path. Suppose ϕ is the query asked by N on the path of computation defined by w.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:

➢ If $a_1 = 1$ and $\phi(u_1) = 1$, M continues the simulation; else it stops and outputs 0. (In this case, M ignores v_1.)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:

➢ If $a_1 = 0$ and $\phi(v_1) = 0$, M continues the simulation; else it stops and outputs 0. (In this case, M ignores u_1.)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- Think of a TM M that takes input x and $w \in \{0,1\}^{q(|x|)}$, $a_1 \in \{0,1\}$ and u_1, $v_1 \in \{0,1\}^{q(|x|)}$, where $q(|x|)$ is the runtime of N on input x, and does the following:
- At the end of the simulation, M outputs whatever N outputs. Note: M is a poly-time TM.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- Obs. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,

➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x ⟺

$\exists u_1 \in \{0,1\}^{q(|x|)}$ $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w, a_1,u_1,v_1) = 1$.

(…will prove the observation shortly. Let's finish the proof.)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- $x \in L \quad \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$ s.t
  - N on computation path w gets answer a_1 from the SAT oracle and accepts $x \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$

$\exists u_1 \in \{0,1\}^{q(|x|)} \;\; \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = \mathrm{NP}^{\mathrm{SAT}}$.
- Proof. Let L be a language in $\mathrm{NP}^{\mathrm{SAT}}$. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- $x \in L \quad \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$ s.t
  - ➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$

$$\exists u_1 \in \{0,1\}^{q(|x|)} \ \forall v_1 \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x,w,\underbrace{a_1,u_1}_{\text{Call it } u},v_1) = 1.$$

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- Special case: N asks at most one query to SAT oracle on every computation path on input x.
- $x \in L \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)}, a_1 \in \{0,1\}$ s.t
  - ➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

    $\exists u \in \{0,1\}^{2q(|x|)+1} \ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,u,v_1) = 1$.
- Therefore, L is in Σ_2 .

Proof of the observation

- Obs. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
  - N on computation path w gets answer a_1 from the SAT oracle and accepts x ⟺

    $\exists u_1 \in \{0,1\}^{q(|x|)}$ $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.
- Proof.(⇒) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 1$, $\exists u_1 \in \{0,1\}^{q(|x|)}$ $\phi(u_1) = 1$ and N accepts x.

Proof of the observation

- Obs. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
- ➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x $\Leftrightarrow$

  $\exists u_1 \in \{0,1\}^{q(|x|)}$ $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.
- Proof.($\Rightarrow$) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 1$, $\exists u_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

In this case, M ignores v_1.

Proof of the observation

- Obs. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
- N on computation path w gets answer a_1 from the SAT oracle and accepts x ⟺

  $\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

- Proof.(⇒) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 0$, $\forall v_1 \in \{0,1\}^{q(|x|)}\ \phi(v_1) = 0$ and N accepts x.

Proof of the observation

- Obs. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,
- N on computation path w gets answer a_1 from the SAT oracle and accepts x ⟺

  $\exists u_1 \in \{0,1\}^{q(|x|)}$ $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

- Proof.(⇒) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- If $a_1 = 0$, $\forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

In this case, M ignores u_1.

Proof of the observation

- Obs. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,

➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x ⟺

$$\exists u_1 \in \{0,1\}^{q(|x|)} \ \forall v_1 \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x,w,a_1,u_1,v_1) = 1.$$

- Proof.(⟹) M simulates N on computation path w. Let ϕ be the query asked by N to SAT.
- Irrespective of the value of a_1,

$$\exists u_1 \in \{0,1\}^{q(|x|)} \ \forall v_1 \in \{0,1\}^{q(|x|)} \text{ s.t. } M(x,w,a_1,u_1,v_1) = 1.$$

Proof of the observation

- Obs. For any $w \in \{0,1\}^{q(|x|)}$ and $a_1 \in \{0,1\}$,

➢ N on computation path w gets answer a_1 from the SAT oracle and accepts x ⇔

$\exists u_1 \in \{0,1\}^{q(|x|)}\ \forall v_1 \in \{0,1\}^{q(|x|)}$ s.t. $M(x,w,a_1,u_1,v_1) = 1$.

- Proof.(⇐) Need to show that N on computation path w gets answer a_1 from the SAT oracle. (*Homework*)

Oracle definition of Σ_i

- Theorem. $\Sigma_2 = NP^{SAT}$.
- Proof. Let L be a language in NP^{SAT}. There's a NTM N that decides L with oracle access to SAT.
- General case: N asks at most $q(|x|)$ queries to SAT oracle on every computation path on input x.
- *Homework*: Prove the general case. Define the poly-time machine M appropriately.

Oracles versus efficient algorithms

- Definition. A language L is in P^{SAT} if there is a poly-time TM with oracle access to SAT that decides L.
- $\Delta_2 := P^{SAT} \subseteq \Sigma_2 \cap \Pi_2$.
- A SAT oracle gives us the ability to solve SAT efficiently "much like" a poly-time algorithm for SAT.

Oracles versus efficient algorithms

- Definition. A language L is in P^{SAT} if there is a poly-time TM with oracle access to SAT that decides L.
- $\Delta_2 := P^{SAT} \subseteq \Sigma_2 \cap \Pi_2$.
- A SAT oracle gives us the ability to solve SAT efficiently "much like" a poly-time algorithm for SAT.

- Yet, in the first case we believe $P^{SAT} \neq NP^{SAT}$, whereas in the second case PH collapses to P, i.e., $P^{SAT} = NP^{SAT}$.
- Why? Think to understand the difference between oracles and poly-time algorithms for SAT (*Homework*).